

第 1 章 Java 的开发运行环境

学好 Java 最重要的一个步骤就是上机编程，熟悉 Java 的开发运行环境是成为 Java 程序员的第一步。本章将详细介绍如何安装并配置好 Sun 公司提供的 JDK1.5 for Windows，如何编制一个简单的 Java 程序，如何编译一个 Java 源程序，如何运行编译好的 class 文件以及如何避免初学者常犯的错误。通过本章的学习，将轻松地迈入 Java 的殿堂。

1.1 Java 的运行环境与虚拟机

任何一个可执行文件，都必须在某个平台上才能运行。例如，Windows 下的 exe 文件，必须在 Windows 环境下、X86 硬件平台上才能运行。这些 exe 文件，通常是使用 C/C++、Pascal 或 VB 等语言编程，然后通过编译、链接而形成的。在这些可执行文件中，包含了运行它的硬件平台的相关信息，所以如果要把它移植到其他平台上，必须要重新编译，甚至要修改源文件。

Java 和这些语言不同，它的最大特点就是平台无关性。Java 文件经编译后，生成的是一个后缀名为 class 的文件。这是一种字节码文件，它不像普通可执行文件那样包含硬件信息，而是完全与硬件平台无关，也就是无法直接由操作系统调用运行。所以 Java 的应用程序，需要一个更为复杂的平台才能运行。这个运行平台，包括计算机操作系统、适配器、Java 虚拟机、Java 基本软件和 Java 应用程序接口，它们负责将 Java 的字节码翻译成硬件可以接受的指令。整个运行系统的结构如图 1.1 所示。

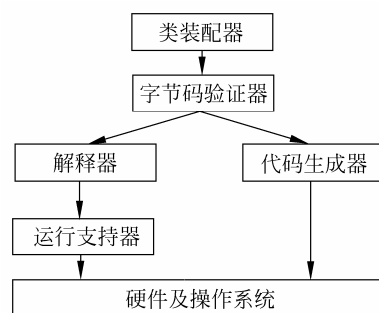


图 1.1 Java 运行系统

Java 运行系统执行 Java 应用程序的过程包括三个步骤：首先是代码的装入，然后是代码的验证，最后是代码的执行。

代码的装入由类装配器完成，它主要负责装入程序运行时所需要的全部代码，包括程序运行中调用到的其他类。当类装配器装入一个类后，该类被放在自己的命名空间中，除了通过符号引用其他类之外，该类不能影响其他类所在的空间。当装入了运行程序所需要的所有类后，运行系统就能确定整个可执行程序内存的布局。


然后，被装入的代码将由字节码验证器进行安全检查，以确保字节代码不存在违反访问权限、不规范的数据类型和非法调用等问题。

通过校验之后，代码就可以运行了。Java 的字节码有两种运行方式：

- ❑ 解释执行方式：通过“解释器”，将字节码翻译成机器码，然后由“运行支持库”将机器码送往硬件执行。整个执行过程是一边翻译一边执行，称为解释执行。Java 系统一般采用这种方式。
- ❑ 即时编译方式：通过“代码生成器”，先一次性地将所有字节码翻译成适用于特定计算机系统的机器码，然后送往硬件执行。对程序运行速度要求较高时，一般采用这种方式。

从图 1.1 可以看出，Java 的字节码并没有直接运行在硬件平台上，而是在一个虚拟的软件平台上运行。这个虚拟的软件平台，被称为 Java 虚拟机 (Java Virtual Machine, JVM)。为了让编译产生的字节码可以更好地解释与执行，通常把 JVM 分成 6 个功能模块：JVM 解释器、指令系统、寄存器、栈、存储区和碎片回收区。

- ❑ JVM 解释器：JVM 解释器负责将字节码转换成为 CPU 能执行的机器指令。
- ❑ 指令系统：指令系统同硬件计算机很相似。一条指令分成操作码和操作数两部分。操作码为 8 位二进制数，操作数可以根据需要而定。操作码是为了说明一条指令的功能，所以 JVM 可以有多达 256 种不同的操作指令。
- ❑ 寄存器：JVM 有自己的虚拟寄存器，这样就可以快速地和 JVM 的解释器进行数据交换。为了实现必需的功能，JVM 设置了 4 个常用的 32 位寄存器：pc（程序计数器）、optop（操作数栈顶指针）、frame（当前执行环境指针）和 vars（指向当前执行环境中第一个局部变量的指针）。
- ❑ 栈：JVM 栈是指令执行时数据和信息存储的场所和控制中心，它提供给 JVM 解释器运算时所需要的信息。
- ❑ 存储区：JVM 存储区用于存储编译后的字节码等信息。
- ❑ 碎片回收区：JVM 碎片回收，是指将那些使用后的 Java 类的具体实例从内存中进行回收。因此，可以避免开发人员自己编程控制内存的麻烦。随着 JVM 的不断升级，其碎片回收技术和算法也更加合理。比较经典的算法有引用计数、复制、标记-清除和标记-整理。在 JVM 1.4.1 版以后，产生了一种代收集技术。简单地说，就是利用对象在程序中生存的时间划分成代，以这个代为标准进行碎片回收。

说明：JVM 的运用，真正让 Java 实现了“一次编译，处处运行”，它是整个运行系统的核心。

1.2 Java 的开发环境

要开发 Java 程序，必须要有一个开发环境。而一提到开发环境，读者可能首先想到的就是那些大名鼎鼎的集成开发工具：Eclipse、JBuilder、Microsoft Visual J++、JCreator Pro、Net Beans、Sun Java Studio、Visual Age for Java、WebLogic Workshop、Visual Cafe for Java、IntelliJ 等。但实际上，如此众多的开发工具中，除了 Microsoft Visual J++ 是使用自己的编译器外，其余的大多是使用 Sun 公司提供的免费的 JDK 作为编译器，只不过是开发了一个集成环境套在外面，方便程序员编程而已。


这些集成开发工具，虽然方便了程序员开发大型软件，但是它们封装了很多有关 JDK (Java Development Kit) 的基本使用方法，在某些方面又过于复杂，并不太适合初学者使用。因此，在本书中，将首先介绍最基本的 JDK 的安装和使用。

1.2.1 JDK 的安装

JDK 是 Sun 公司发布的免费的 Java 开发工具。它从 Alpha1.0 开始，先后经历了 JDK1.0、JDK1.1…多次升级，目前最新版本是 2006 年底发布的 JDK1.6。

按照应用平台进行划分，JDK 有三个主要成员：可扩展的企业级应用 Java 2 平台的 J2EE (Java 2 Enterprise Edition)、用于工作站和 PC 机的 Java 标准平台的 J2SE (Java 2 Standard Edition)，以及用于嵌入式消费电子平台的 J2ME (Java 2 Micro Edition)。

按照运行的操作系统进行划分，JDK 分别有 for Windows、for Linux、for Solaris 和 for MacOS 等不同版本。

 **说明：**本书中使用的是 J2SE 平台的 JDK1.5 for Windows。由于 JDK 是向下兼容的，后继版本也可以编译运行本书中的示例程序。

各个版本的 JDK 安装和配置过程并无多大的差异。下面就以 JDK1.5 为例，来介绍它的安装和配置。

开始安装前，读者需要到 Sun 公司或者是其他相关网站下载 JDK1.5 for Windows。JDK1.5 自推出后经过了多次升级，本书使用的是 Update 4 版，该版本 BUG 很少，运行相当稳定。下载的文件名为 jdk-1_5_0_04-windows-i586-p.exe，这是一个普通的 Windows 下的可执行文件，可以安装在 Windows98 及其以后所有版本的 Windows 平台上。双击该文件，就可以开始安装了。首先是自解压过程，用户不必干涉，稍作等待，当自解压过程完成后，将出现需要用户选择的步骤。

(1) 接受许可证协议

在图 1.2 中必须选择接受许可证协议，否则无法继续安装。选中“我接受该许可证协议中的条款 (A)”之后，再单击“下一步”按钮。

(2) 选择要安装的模块和路径

这是安装中最重要的一步，如图 1.3 所示。



图 1.2 选择是否接受许可证

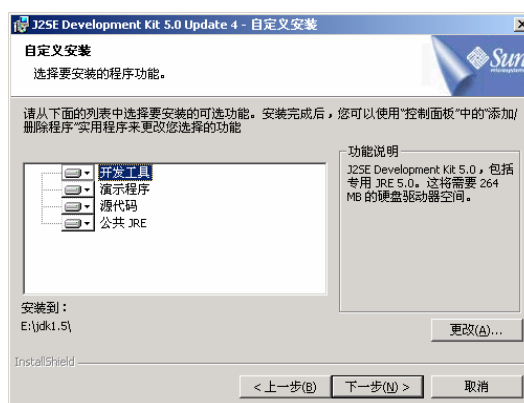


图 1.3 选择要安装的模块和路径

其中，“开发工具”是必选的，“演示程序”和“源代码”是给开发者做参考的，除非硬盘空间非常紧张，否则最好安装。“公共 JRE”是一个独立的 Java 运行时环境（Java Runtime Environment, JRE）。任何应用程序均可使用此 JRE。它会向浏览器和系统注册 Java 插件和 Java Web Start。如果不选择此项，IE 浏览器可能会无法运行 Java 编写的 Applet 程序。

安装路径是默认安装到 C 盘。如果需要更改此路径，单击“更改”按钮，选择你想要安装的盘符，并填入文件夹名称，笔者将它安装到 E:\jdk1.5 下面。读者一定要记录下自己的安装路径，因为稍后对 JDK 进行配置时，将用到此路径。

（3）查看安装进度

在图 1.3 中配置好后，再单击“下一步”按钮，出现如图 1.4 所示的对话框，用户可以看到安装的进度。

这个过程不需要用户干涉。直到安装快完毕时，如果用户前面选择了安装公共 JRE，则会进入到第（4）步，要求用户选择安装 JRE 的模块和安装路径。否则，将直接进入第（6）步。

（4）选择 JRE 的安装模块和路径

图 1.5 显示了安装 JRE 时的模块选项。其作用主要是用于对欧洲语言的支持。通常情况下，不需要用户修改这些默认选项。默认的安装路径是 C 盘，也可以不用修改。直接单击“下一步”按钮即可。

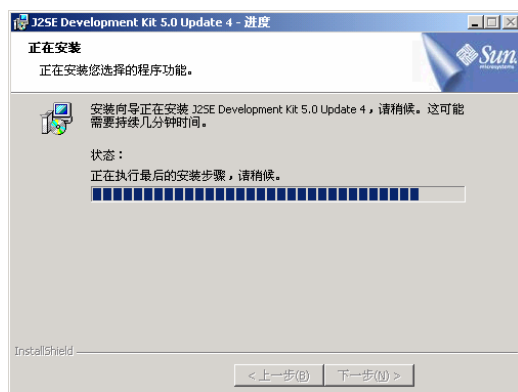


图 1.4 JDK 安装进度

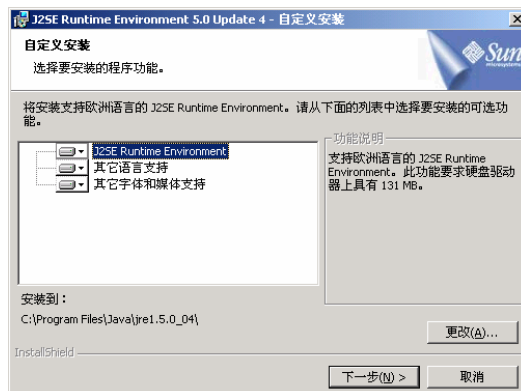


图 1.5 选择安装 JRE 的模块和路径

(5) 为浏览器安装 JVM

IE 浏览器从 5.0 起，不再捆绑 JVM。图 1.6 显示了向 IE 浏览器注册 JVM 插件，如果这里不选择注册，IE 浏览器可能无法运行嵌入在网页中的 Applet 程序。如果读者需要开发 Applet 程序，那么这里不要更改它的默认值。

(6) 结束安装

如果前面一切正常，将出现如图 1.7 所示的对话框，表示 JDK1.5 已经正确安装到机器上。读者只要单击“完成”按钮，则安装过程到此结束。




图 1.6 为浏览器注册 JVM 插件



图 1.7 安装完成

JDK 安装完毕后，在安装路径下有以下几个文件夹。

- ❑ bin 文件夹：存放编程所要用到的开发工具，包括编译器、解释执行程序、小应用程序浏览器、调试器、文档生成工具、反编译器等。
- ❑ demo 文件夹：存放演示程序，开发者可以从中学习如何使用 Java 的类库。
- ❑ include 文件夹：存放本地文件（Native means）。
- ❑ jre 文件夹：Java 运行时环境的根目录，存放 JVM 所需的各种文件。
- ❑ lib 文件夹：存放库文件。
- ❑ sample 文件夹：类似于 demo 文件夹，也是存放 Java 的源程序，该程序多数和网络相关。

 **注意：**和一般的 Windows 程序不同，JDK 安装成功后，不会在“开始”菜单和桌面生成快捷方式。这是因为 bin 文件夹下面的可执行程序都不是图形界面的，它们必须在控制台中以命令行方式运行。另外，还需要用户手工配置一些环境变量才能正常使用 JDK。

1.2.2 如何设置系统环境变量

环境变量是包含关于系统及当前登录用户的环境信息的字符串，一些程序使用此信息确定在何处放置和搜索文件。和 JDK 相关的环境变量有两个：path 和 classpath。其中，path 环境变量告诉操作系统到哪里去查找 JDK 工具，classpath 环境变量则告诉 JDK 工具到哪

里去找类文件（.class 文件）。

对于 Windows9.X，设置环境变量要修改 autoexec.bat 文件。如果是 Windows 2000 和 Windows XP，则需要通过系统属性来修改。假设 JDK 安装在 e:\jdk1.5 下面，分别介绍如何在以上两种系统下设置环境变量。

1. 在Windows 9.X下设置环境变量

在 C 盘根目录下找到 autoexec.bat 文件，右击此文件。在右键菜单中选择“编辑”项，就可以用记事本的形式打开此文件，然后在文件的最末尾添加下面两行内容：

```
set path=%path%;e:\jdk1.5\bin;
set classpath=%classpath%;e:\jdk1.5\lib;.
```

其中，set 是 DOS 中的一个内部命令，用于设置环境变量。%path%表示前面已经设置好的环境变量 path 的值。e:\jdk1.5 就是 JDK 安装的文件夹，读者的 JDK 如果没有安装在此处，需要将这个值替换成自己安装 JDK 的位置。

注意：环境变量 classpath 那一行的末尾，有一个“.”，它表示当前工作目录，一定不能漏掉。

修改完成并存盘后，重新启动计算机或者双击 autoexec.bat 文件使其运行，然后就可以使用 JDK 了。

2. 在Windows 2000和Windows XP中设置环境变量

在 Windows 2000 和 Windows XP 中设置环境变量的方法是完全一样的，基本原理与 Windows9.X 下的相同，只是设置的位置不同。设置方法为：在“开始”主菜单中选择“设置”→“控制面板”→“系统”→“高级”→“环境变量”选项，找到如图 1.8 所示的位置。读者的系统如果是 Windows XP，而且控制面板不是经典视图而是分类视图时，设置步骤中在“控制面板”和“系统”之间增加一步“性能和维护”即可。

图 1.8 中“liuxin 的用户变量”设置之后只对该用户有效，而“系统变量”则对所有用户有效。建议读者对用户变量进行操作。如果原环境变量中没有变量 classpath，则单击“新建”按钮，进入如图 1.9 所示的对话框中进行设置。在图 1.8 中，如果已经有了 classpath，则选中它，单击“编辑”按钮，在图 1.9 所示的“变量值”文本框加入“e:\jdk1.5\lib;.”然后单击“确定”按钮即可。

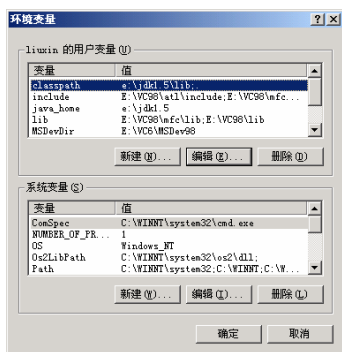


图 1.8 设置环境变量

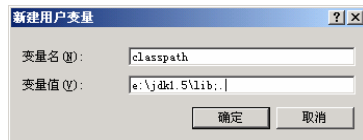


图 1.9 设置环境变量 classpath

环境变量 `path` 的设计与 `classpath` 类似, 如果原环境变量中没有该变量, 则新建它, 如果有该变量则编辑它, 如图 1.10 所示。

两个环境变量设置完成后, 在图 1.8 中单击“确定”按钮就可以保存设置, 然后开始使用 JDK。初学者最容易犯的错误就是环境变量设置错误, 这样当运行 Java 程序时, 总是会出现下面这样的提示:

```
Exception in thread "main" java.lang.NoClassDefFoundError: ×××
```

如果确认程序没有错误, 就需要检查环境变量是否设置正确。除了按照上面的步骤重新检查设置之外, 也可以在 DOS 窗口中进行检查。具体方法是在“开始”菜单处单击“运行”命令, 打开“运行”对话框, 在“打开”输入框中输入 `cmd` (Windows 98 中是输入 `command`), 然后单击“确定”按钮, 如图 1.11 所示。

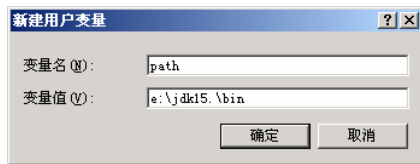


图 1.10 设置环境变量 `path`

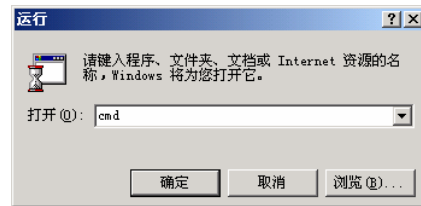


图 1.11 运行 `cmd` 命令

进入 DOS 窗口后, 在命令行输入命令: `set`, 可以看到如图 1.12 所示的环境变量信息。

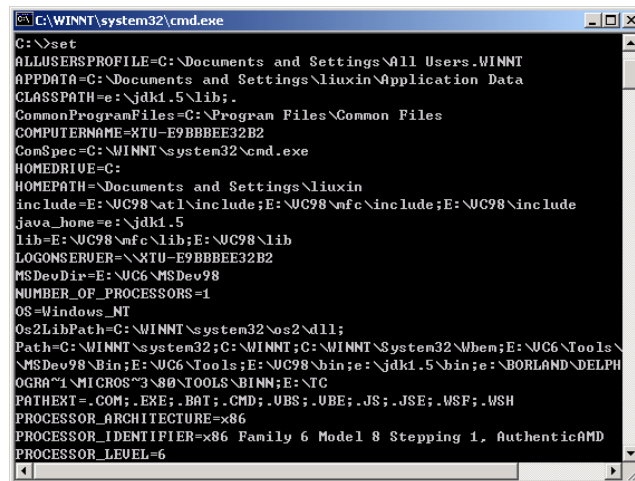


图 1.12 显示环境变量

图 1.12 显示的是本机所有的环境变量, 仔细查找其中的 `path` 和 `classpath` 所在行, 看是否设置正确。只有当环境变量设置正确后, 才能正常使用 JDK。

1.2.3 编译命令的使用

JDK 中所有的命令都集中在安装目录的 `bin` 文件夹下面, 而且都是控制台程序, 要以

命令行方式运行。基本步骤和 1.2.2 小节中运行 set 命令类似，只是最后输入的命令不是 set，而是相应的程序名称。

JDK 所提供的开发工具主要有编译程序、解释执行程序、调试程序、Applet 执行程序、文档管理程序、包管理程序等。下面介绍最常用的编译程序和解释执行程序的使用方法。

JDK 的编译程序是 javac.exe，该命令将 Java 源程序编译成字节码，生成与类同名但后缀名为.class 的文件。编译器会把.class 文件放在和 Java 源文件相同的一个文件夹里，除非用了-d 选项。如果引用到某些自己定义的类，必须指明它们的存放路径，这就需要利用环境变量参数 classpath。注意，它总是将系统类的目录默认地加在 classpath 后面，除非用 -classpath 选项来编译。javac 的一般用法如下：

```
javac [-选项] file.java...
```

其中，file.java 是要编译的源文件，这是必须要输入的，[-选项]为可选项，javac 中的编译选项及其含义如表 1.1 所示。


表 1.1 javac中的编译选项及其含义

选 项	含 义
-g	生成所有调试信息
-g:none	不生成任何调试信息
-g:{lines,vars,source}	只生成某些调试信息
-nowarn	不生成任何警告
-verbose	输出有关编译器正在执行的操作的消息
-deprecation	输出使用已过时的 API 的源位置
-classpath <路径>	指定查找用户类文件的位置
-cp <路径>	指定查找用户类文件的位置
-sourcepath <路径>	指定查找输入源文件的位置
-bootclasspath <路径>	覆盖引导类文件的位置
-extdirs <目录>	覆盖安装的扩展目录的位置
-endorseddirs <目录>	覆盖签名的标准路径的位置
-d <目录>	指定存放生成的类文件的位置
-encoding <编码>	指定源文件使用的字符编码
-source <版本>	提供与指定版本的源兼容性
-target <版本>	生成特定 VM 版本的类文件
-version	版本信息
-help	输出标准选项的提要
-X	输出非标准选项的提要
-J<标志>	直接将 <标志> 传递给运行时系统

虽然 javac 的选项众多，但对于初学者而言，并不需要一开始就掌握这些选项的用法，只需要掌握一个最简单的用法就可以了。比如生成一个 hello.java 文件，要执行它，只需在命令行输入：

```
c:\>javac hello.java
```


这里的 `hello.java` 是准备编译的文件名,这里必须将文件名完整输入,不能省略后缀名。如果编译成功,它不会有任何提示,因为 Java 遵循的原则是“没有消息就是好消息”,并且会在 `hello.java` 所在的同一文件夹下生成一个或多个 `.class` 文件。

 **注意:** 这个 `class` 文件的主文件名并不一定和 `hello.java` 的主文件同名,它的名称会和源程序中定义的类的名字相同。

编译成功后,下一步就是运行这个 `class` 文件,这需要用到解释执行命令。

1.2.4 解释执行命令的使用

JDK 的解释执行程序是 `java.exe`,该程序将编译好的 `class` 加载到内存,然后调用 JVM 来执行它。它有两种用法:

```
执行一个 class 文件:  java [-选项] class [参数...]
执行一个 jar 文件:   java [-选项] -jar jarfile [参数...]
```

关于 `jar` 文件,将在 4.14 节介绍。注意上面命令中的[参数...],表示要传递给执行文件的参数,称为“命令行参数”,它的详细用法将在 3.8 节介绍。Java 命令中的选项及其含义如表 1.2 所示。

表 1.2 Java命令中的选项及其含义

选 项	含 义
<code>-client</code>	选择客户虚拟机(这是默认值)
<code>-server</code>	选择服务虚拟机
<code>-hotspot</code>	与 <code>client</code> 相同
<code>-cp <class search path of directories and zip/jar files></code>	用分号分隔的一系列文件的搜索路径
<code>-classpath <class search path of directories and zip/jar files></code>	与 <code>cp</code> 相同
<code>-D<name>=<value></code>	设置系统属性
<code>-verbose[:class gc jni]</code>	开启详细输出
<code>-version</code>	输出产品的版本然后退出
<code>-version:<value></code>	指定要特定版本才能运行
<code>-showversion</code>	输出产品的版本然后继续运行
<code>-jre-restrict-search -jre-no-restrict-search</code>	在版本搜索中包含/排除用户私有的 JRE
<code>-? -help</code>	显示帮助信息
<code>-X</code>	显示非标准选项的帮助
<code>-ea[:<packagename>...[:<classname>]]</code>	开启断言
<code>-enableassertions[:<packagename>...[:<classname>]]</code>	与 <code>ea</code> 相同
<code>-da[:<packagename>...[:<classname>]]</code>	关闭断言
<code>-disableassertions[:<packagename>...[:<classname>]]</code>	与 <code>da</code> 相同
<code>-esa -enablesystemassertions</code>	开启系统断言
<code>-dsa -disablesystemassertions</code>	关闭系统断言
<code>-agentlib:<libname>[=<options>]</code>	装载本地代理库
<code>-agentpath:<pathname>[=<options>]</code>	装载指定了全路径的本地代理库
<code>-javaagent:<jarpath>[=<options>]</code>	装载 Java 程序的语言代理

与 `javac` 相同，初学者只要掌握最简单的用法就可以了。比如上例中，生成了一个 `hello.class` 的文件。要执行它，只需要在命令行输入：

```
c:\>java hello
```

⚠注意：java 命令是区分大小写的。大小写不同，表示不同的文件。所以文件名 `hello` 一定不能写成 `Hello` 或 `HELLO` 等。还有，文件的后缀 `.class` 也不能要，只要主文件名就可以了。具体原因，将在 4.13 节中说明。

限于篇幅，不能一一介绍 JDK 中所有的命令。读者如果想要详细了解这些命令的使用，可以查阅 Sun 公司发布的 JDK 5.0 Documentations 中的 JDK Tools and Utilities 部分。也可以直接在命令行输入想要执行的程序名，可以看到一个简要的帮助。

1.2.5 UltraEdit 的使用

Java 的源程序必须以纯文本文件的形式编辑、保存。而在 JDK 中，并没有提供文本编辑器。用户编辑源程序时，需要自行选择文本编辑器。最简单的纯文本编辑器是 Windows 自带的记事本。但是记事本不仅功能太弱，而且在作为 java 的源程序编辑器时，存盘时特别容易出错，如图 1.13 所示。

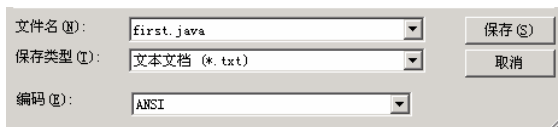


图 1.13 错误地用记事本保存源文件

⚠注意：记事本的默认类型是文本文档。即使用户像图 1.13 中那样输入文件名为“`first.java`”，则保存之后，它的文件名仍然会变成“`first.java.txt`”，编译时将找不到源文件。这里必须在“保存类型”下拉列表框中选择“所有文件”，如图 1.14 所示。



图 1.14 正确地用记事本保存源文件

正因为记事本存在诸多不足，所以笔者推荐使用功能更为方便的文本编辑器——UltraEdit 作为学习 Java 过程中的编辑工具。UltraEdit 是 Windows 下功能最强大的纯文本编辑器，读者可以到华军软件园或是其他网站下载，有英文版和汉化版两个版本。作为源程序编辑器时，它有 3 个很有特色的功能：

- ❑ 支持语法高亮度显示（也就是 Java 等语言的关键字用不同的颜色显示出来）。
- ❑ 可以执行 DOS 命令。
- ❑ 可同时编辑多个文件，且每个文件大小不限。但在某一时刻，只有最前面的文件为活动文件。

有了这几个功能，用户可以将 UltraEdit 搭建为一个简单的集成编程环境，所以很多程序员将它作为小程序开发工具。当然也可以将 UltraEdit 作为开发 Java 程序的集成环境，下面来看看一些使用 UltraEdit 的关键步骤。

1. 新建和编辑源程序

启动 UltraEdit 后，它会自动建立一个空白文档，也可以选择菜单中的 File→New 命令来新建一个空白文档。用户可以在此编辑自己的 Java 程序，编辑时的各种操作和记事本的使用完全相同，如图 1.15 所示。

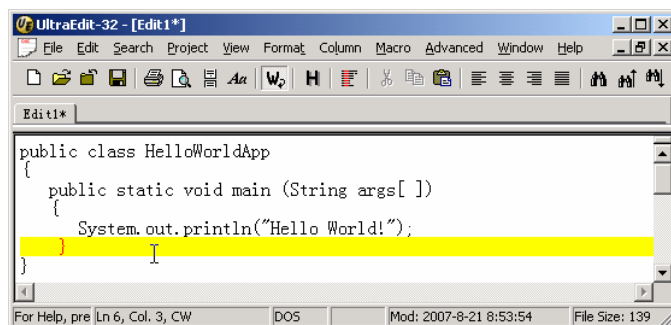


图 1.15 在空白文档中编辑一个源程序

注意：图 1.15 中文档上部的标签，显示为“Edit1*”，说明该文件还从来没有命名保存过。

2. 保存源程序

文档编辑后，需要保存。这要选择菜单中的 File→Save 命令，将出现如图 1.16 所示的对话框。

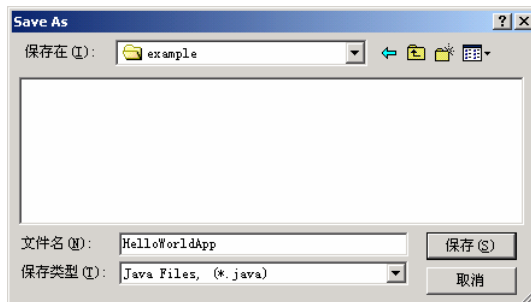


图 1.16 保存源程序

注意：图 1.16 中，选择的保存类型是“Java Files”，这样文件名可以只要主文件名，而无需扩展名。如果没有选择保存类型为“Java Files”，而是其他任意类型，则文件名必须是“主文件名.java”的形式。

保存成功后,就可以看到源程序中各种关键字、数字、字符串等都以不同的颜色显示,这就是 UltraEdit 的“语法高亮度”功能。

3. 编译源程序

要编译源程序,不必再运行 cmd 转到 DOS 窗口,可以直接在 UltraEdit 中编译。方法是选择菜单中的 Advance→Dos Command 命令,出现如图 1.17 所示的对话框,在其中填入所需要的编译命令。

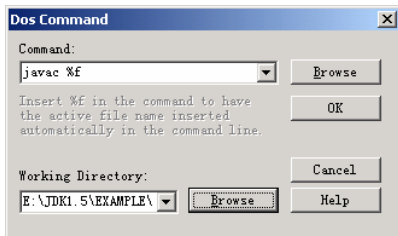


图 1.17 编译源程序

注意,图 1.17 中的 Command 下拉列表框中,填入的是“javac %f”。“javac”是前面介绍过的编译命令;“%f”是 UltraEdit 自己定义的一个宏代换变量,表示当前正在编辑的文件全名。例如,前面保存的文件名为“HelloWorldApp.java”,那么 UltraEdit 就会把“javac %f”代换为“javac HelloWorldApp.java”,这正是 JDK 的编译命令。Working Directory 下拉列表框中,填入的是“E:\JDK1.5\EXAMPLE”,这是当前文件存放的位置。笔者建议读者在学习过程中,将编制的所有 java 源程序都集中存放在一个文件夹下,这样便于管理。

填好这两个下拉列表框之后,单击 OK 按钮,UltraEdit 就会执行 command 框中的命令。而且会自动保存这些命令,以后用户无需再重新填写,只要调出该对话框就可以重复执行上面的命令。

执行编译命令后,UltraEdit 会接收 javac 返回的信息,并显示在一个名为“Command Outputx”的文档中供用户查看。如果该文档为空,表示编译已经成功,用户可以按 Ctrl+F4 键将该文档关闭。否则就需要修改源程序,直到编译通过为止。

4. 解释执行程序

源程序编译通过后,就可以执行编译好的 class 文件,这同样可以利用 UltraEdit 来执行。方法是单击菜单中的 Advance→Dos Command 命令,出现如图 1.18 所示的对话框,在其中填入所需要的执行命令。

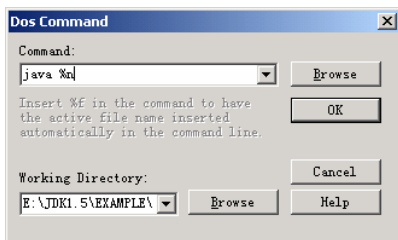



图 1.18 执行程序

注意，图 1.18 中的 Command 下拉列表框中，填入的是“java %n”。“java”是前面介绍过的解释执行命令；“%n”是 UltraEdit 定义的另一个宏代换变量，表示当前正在编辑的主文件名。比如，前面保存的文件名为“HelloWorldApp.java”，编译生成后的 class 文件为“HelloWorldApp.class”，那么 UltraEdit 就会把“java %n”代换为“java HelloWorldApp”，这正是 JDK 的解释执行命令。另外一个关于 Working Directory 下拉列表框中的内容无须改变。单击 OK 按钮，程序就会被执行。

程序执行后输出的结果，UltraEdit 同样会将它接收后显示在一个名为“Command Outputx”的文档中供用户查看。

 **注意：**上面所述的第（3）、（4）步操作中，准备编译或执行的文件必须是当前活动文件。如果不是，只要简单地单击文件对应的标签，就可将它置为活动文件。以上讲解的时候，是以英文版的 UltraEdit 为标准。如果读者下载的版本是汉化版，操作步骤也是完全相同的，只是上述这些图片中显示的信息为对应的汉语。

1.3 一个简单的 Java 应用程序

Java 的程序有两类：一类是只能嵌入在网页文件中，通过浏览器运行的程序，被称为 Applet，译为小应用程序；除此之外的 Java 程序，都被称为 Application，译为应用程序。有了前面的基础，就可以开始编制自己的第一个程序了。本节介绍一个最简单的应用程序的编制。

【例 1.1】 编程输出字符串：Hello, World!


//-----文件名 HelloWorldApp.java, 程序编号 1.1-----

```
public class HelloWorldApp
{
    public static void main (String args[ ])
    {
        System.out.println("Hello World!");
    }
}
```

- ❑ 程序中，首先用保留字 class 来声明一个新的类，其类名为 HelloWorldApp，该类是一个公共类（public）。整个类的定义由大括号{}括起来。
- ❑ 在该类中定义了一个 main()方法，其中，public 表示访问权限，表明所有的类都可以使用本方法。
- ❑ static 指明该方法是一个类方法（又称静态方法），它可以通过类名直接调用。
- ❑ void 指明 main()方法不返回任何值。对于一个应用程序来说，main()方法是必需的，而且必须按照如上的格式来定义。Java 解释器以 main()为入口来执行程序，main()方法定义中，括号中的“String args[]”是传递给 main()方法的参数。
- ❑ 在 main()方法的实现（大括号）中，只有一条语句：System.out.println ("Hello World!");，它用来实现字符串“Hello world!”的输出。

- ❑ `System.out.println()`方法是最常用的输出方法，括号内的参数一般是一个字符串，也可以是后面要介绍的各种数据类型。如果有多个输出项目，用“+”将它们连接起来。

该源程序可以用前面介绍的记事本或 UltraEdit 来编辑。

 **注意：**Java 源程序是区分大小写的，例如该程序中的 `String` 和 `System` 两个单词都必须以大写字母开头，请不要输错，否则编译将无法通过。

存盘的时候，文件名也是区分大小写的。Java 规定，如果类前面用 `public` 来修饰，那么文件名必须和类名完全相同。笔者建议无论类前面是否有 `public` 修饰，文件名也应与类名相同，而且在一个源程序中，只定义一个类。这么做不仅便于编程时使用 UltraEdit 来编译运行，也便于以后对源程序进行修改和维护。

像上面这个程序 1.1，它的类名为 `HelloWorldApp`，笔者就将源文件命名为 `HelloWorldApp.java`，并保存在 `e:\jdk1.5\example` 下面（后面如果不做特殊说明，所有 Java 源程序和生成的 class 文件都存放在此文件夹下）。

存盘之后，就可以编译运行程序了。如果使用记事本，则需要经过如下几个步骤：

- (1) 运行 `cmd` 命令，进入到 DOS 窗口。
- (2) 执行 DOS 命令，进入 E 盘。

```
C:\Documents and Settings>e:
```

- (3) 进入到 `e:\jdk1.5\example` 文件夹下面。

```
e:\>cd jdk1.5\example
```

- (4) 使用编译命令编译源程序。

```
e:\jdk1.5\exmaple>javac HelloWorldApp.java
```

如果有错误提示，请仔细检查源程序编辑是否正确。初学者很容易犯一些极不起眼的小错误。例如，漏写一个分号，写错一个字母，都会导致编译失败。如果没有提示，则表示编译成功。编译生成的是一个名为 `HelloWorldApp.class` 的文件。这里的主文件名“`HelloWorldApp`”，是根据源程序中类的名字“`HelloWorldApp`”生成的，与 `HelloWorldApp.java` 的文件名完全没有关系，读者可以用 `dir` 命令或是资源管理器来查看是否生成了该文件。

- (5) 执行 `java` 命令，运行程序。

```
e:\jdk1.5\exmaple>java HelloWorldApp
```

 **注意：**`HelloWorldApp` 的大小写不能错。

- (6) 如果一切顺利，将在屏幕上显示一行字符串：

```
Hello, World!
```

(7) 如果没有上面这行字符串, 而是这样一行提示:

```
Exception in thread "main" java.lang.NoClass-
DefFoundError: HelloWorldApp
```

通常是因为环境变量设置错误, 或者是输入 java 命令时大小写弄错了, 也有少部分是源于源程序中的 main() 方法的名字或者是参数写错了, 请读者仔细检查。

以上所述的步骤, 就是编制一个程序的一般过程。其中有多步可能会出错, 需要反复修改源程序。越是复杂的程序, 修改源程序的次数就越多, 这也是程序员积累编程经验的过程。整个程序编制的过程如图 1.19 所示。

如果使用 UltraEdit, 则编译和运行步骤不必使用命令行方式, 而可以像 1.2 节中图 1.17 和图 1.18 介绍的那样, 通过对话框来编译和运行, 而且不用考虑文件名的大小写问题, 这样可以大大提高编程的效率。

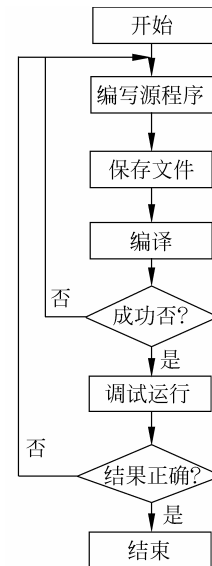


图 1.19 程序编制的过程

1.4 一个简单的 Java 小程序

Applet 程序只能嵌入 HTML 网页中通过浏览器来运行。HTML 是 Hyper Text Markup Language 的缩写, 它是浏览器的通用语言。HTML 是由纯文本字符组成的, 其中有各种各样预定义的标签以及用于显示的文本。浏览器根据这些标签的意义, 将文本按照一定的格式显示出来, 这就是平常看到的网页。一个 HTML 结构一般具有如下形式:

```
<html>
<head>
<title> .....</title>
.....
</head>
<body>
.....
</body>
</html>
```

可以看到, HTML 标签多数是成对出现的, 如<body>...</body>。Applet 程序就嵌入在这些标签中间。当然, 它要用到自己特制的标签<applet>。下面来编制一个简单的 Applet 程序。

【例 1.2】 第一个 Java Applet 程序。

//-----文件名 firstApplet.java, 程序编号 1.2-----

```
import java.applet.*;
import java.awt.*;
```

```
public class firstApplet extends Applet{
    public void paint(Graphics g){
        g.drawString("这是我用 Java Applet 生成的文字!", 150, 25);
    }
}
```

- ❑ 这个程序中没有 main()方法,取而代之的是 paint()方法,这个方法会被浏览器自动调用。这是 Applet 和 Application 程序的根本区别。
- ❑ drawString()方法用于输出信息。
- ❑ import 关键字表示要引入某个包。
- ❑ extends Applet 表示本类是 Applet 的子类。

源程序编制完成后,要保存为 firstApplet.java 文件,并按照 1.3 小节介绍的方法编译。然后再编写下面这个 HTML 文件。

//-----文件名 firstApplet.htm-----

```
<HTML>
<HEAD>
  <TITLE>first Java Applet</TITLE>
</HEAD>
<BODY>
  Here is the output of my program:
  <APPLET CODE="firstApplet.class" WIDTH=250 HEIGHT=25>
  </APPLET>
</BODY>
</HTML>
```

注意<APPLET CODE="firstApplet.class" WIDTH=150 HEIGHT=25>,其中引号中的内容就是刚才生成的 class 文件名,这里就是 firstApplet.class。将这段 html 代码文件保存为 firstApplet.htm,把它和 firstApplet.class 文件保存在同一个文件夹下,代码的编写过程就完成了。

现在可以用 IE 浏览器打开 firstApplet.htm 文件,如果浏览器安装了 JVM,就可以看到显示的效果。如果没有安装 JVM,可以使用 JDK 提供的工具 AppletViewer。在 DOS 窗口中输入:

```
e:\jdk1.5\example>appletviewer firstApplet.htm
```

运行结果如图 1.20 所示。



图 1.20 Applet 运行结果

Sun 公司最初设计 Applet 是为了增强网页的表现能力和与用户的交互能力。早期的 Web 网页几乎全是静止的图片和文字,也无法和用户交互。而 Applet 中可以显示动画、播

放声音，拥有各种控件，可以接收用户输入的数据，执行用户的指令。Applet 一经推出，凭借 Sun 公司网站上一杯热气腾腾的咖啡，一夜之间名声大噪，Java 也随即走红。

不过随着技术的进步，浏览器中很快就出现了 JavaScript/VBScript 这样既易于编写，功能也不错的脚本语言，它们完全可以和用户进行交互。微软也很快在万维网浏览器中实现了 ActiveX 技术，它几乎拥有 Applet 所有的功能。凭借万维网浏览器在市场上的垄断地位，ActiveX 迅速占领了绝大多数市场。微软决定从 IE5.0 起，不再捆绑 JVM，如果用户需要，必须另外下载 JVM，这极大地限制了 Applet 的应用。目前已经很少有网页中再嵌入 Applet，它的用途已经不大，所以本书中的绝大多数例子都以 Application 程序形式提供。

1.5 本章小结

本章介绍了 Java 的入门知识，主要是编辑器和编译器的使用。这里使用的是命令行方式来编译程序，更有助于程序员对于编译器工作状态的掌握。由于这里没有介绍集成开发环境，所以读者可能会觉得有点难。特别是在刚刚开始编译运行的时候，可能会多次出现错误，这多数是由环境变量配置错误造成的，需要读者耐心地找出错误。